OULUN AMMATTIKORKEAKOULU

Anssi Söderena

RESPONSIIVISEN WORDPRESS-TEEMAN KEHITYS

Vaattojarvi.fi

RESPONSIIVISEN WORDPRESS-TEEMAN KEHITYS

Vaattojarvi.fi

Anssi Söderena
Opinnäytetyö
Kevät 2018
Tietojenkäsittely
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma

Tekijä(t): Anssi Söderena
Opinnäytetyön nimi: Responsiivisen WordPress-teeman kehitys
Työn ohjaaja: Jouni Juntunen
Työn valmistumislukukausi ja -vuosi: Kevät 2018 Sivumäärä: 39 + 0

Opinnäytetyön tavoitteena oli toteuttaa helposti hallittavat ja responsiiviset verkkosivut Vaattojärven kyläyhdistykselle, sillä kyläyhdistyksen vanhaa sivustoa oli vaikea ylläpitää itse tehdyn hallintapaneelin kautta. Vanhan sivuston ulkoasu oli vanhentunut ja toimi huonosti mobiililaitteilla, sillä se ei ollut responsiivinen. Uudella sivustolla käytetään WordPress-sisällönhallintajärjestelmää, johon kehitettiin uusi teema käyttäen Bootstrap 4 -kirjastoa responsiivisuuden takaamiseksi.

Opinnäytetyön aikana toteutettiin WordPress-teema, jonka kehityksestä käsitellään tärkeimmät askeleet, kuten WordPressin teeman rakenne, sivupohjahierarkia, vaaditut tiedostot sekä määritykset, sivupohjat, sisällön näyttäminen sivustolla, mukauta-valikon muokkaus sekä käännösfunktiot. Teemaa kehittäessä käydään samalla läpi Bootstrap 4 -kirjaston käyttöönotto WordPressissä ja kirjaston uudet ominaisuudet.

Valmis sivusto on tavoitteiden mukaisesti helppokäyttöinen ja responsiivinen. Uuden teeman toteuttaminen ja tarkempi tutustuminen WordPressin dokumentaatioon opetti paljon WordPress-kehityksestä, josta vielä useiden projektienkin jälkeen oppii aina uutta. Oman teeman kehitys on aikaa vievä projekti, mutta varmistaa sen, että sivusto sisältää kaikki tarvittavat ominaisuudet.

Asiasanat: WordPress, Bootstrap, Sisällönhallinta

ABSTRACT

Oulu University of Applied Sciences
Degree programme in Information Technology

Author(s): Anssi Söderena
Title of thesis: Development of a responsive WordPress theme
Supervisor(s): Jouni Juntunen
Term and year when the thesis was submitted: Spring 2018 Number of pages: 39 + 0

The aim of the thesis was to implement a responsive website for Vaattojärvi village association which could be easily maintained. The older website was hard to maintain as the control panel for changing the site's content was self-made. The layout was outdated and worked poorly on mobile devices as it wasn't responsive. The new website uses WordPress content management system and has a new custom theme developed using Bootstrap 4 library to ensure responsiveness.

Key steps of theme development such as WordPress theme structure, page hierarchy, required files and definitions, templates and displaying content of the site, editing the customize menu and translation functions are gone through during this thesis. Also including the integration of Bootstrap 4 library to the theme and a presentation of the new features.

As in accordance with the objectives, the completed website is easy to use and responsive. Implementing a new theme and getting to know more about WordPress documentation has taught a lot about WordPress development. Even after several WordPress theme development projects there was so much more to learn. Developing your own theme is a time-consuming project, but it ensures that the completed website contains all the necessary features.

Keywords: WordPress, Bootstrap, Content Management

SISÄLLYS

1 JOHDANTO

Nykyisin ei enää riitä, että verkkosivu toimii hyvin työpöytälaitteella, koska mobiililaitteilla verkkoa selaavien määrä on vain kasvanut vuosi vuodelta. Maailman väestöstä 51,2 % selaa Internettiä mobiililaitteella ja mikäli siihen lasketaan vielä tablettitietokoneiden käyttäjät, määrä kasvaa 55,34 %:iin (Statcounter 2018, viitattu 25.5.2018). Tämän takia on tärkeää kehittää verkkosivut, jotka toimivat useilla eri laitteilla näytön koosta ja resoluutiosta riippumatta. Lisäksi responsiivisuus on nykyään erittäin olennainen osa verkkosivustoa, sillä useat ihmiset muodostavat mielipiteen käyttökokemuksistaan yritystä, järjestöä tai henkilöä kohtaan verkkosivujen kautta ja esimerkiksi Google arvioi sivut mobiilinäkymän perusteella ja palkitsee hyvän käyttökokemuksen paremmalla sijoituksella hakutuloksissa (Google 2018, viitattu 25.5.2018).

Jotta verkkosivua ei tarvitse kehittää useaan kertaan eri laitteille, ratkaisuna CSS:n kolmanteen tasoon on kehitetty uusi Flexbox-moduuli, joka mahdollistaa elementtien automaattisen mukautumisen näytön koon mukaan. Lisäksi useat eri kirjastot ja rajapinnat, jotka käyttävät myös Flexbox-moduulia, tarjoavat oman ratkaisunsa responsiivisen kehityksen helpottamiseksi esimerkiksi valmiilla luokilla ja tyylimäärityksillä.

Kaikki verkkosivustojen ylläpitäjät eivät hallitse web-ohjelmointia, joten sisällön päivittäminen suoraan tiedoston koodin sekaan ei ole vaihtoehto, vaan sivustolle täytyy toteuttaa oma hallintapaneeli tai käyttää valmista sisällönhallintajärjestelmää, kuten esimerkiksi WordPressiä, Joomlaa tai Drupalia. Useat ylläpitäjät ovatkin valinneet alustaksi WordPressin ja etsineet siihen joko ilmaisen tai maksullisen teeman, jota voi muokata WordPressin hallintapaneelin kautta. Teeman muokattavuus on tosin teemakohtaista ja kaikki teemat eivät pysty kaikkeen, joten yleensä jokin toiminnallisuus tai ominaisuus jää puuttumaan. Vaihtoehdoiksi jää joko valmiin teeman muokkaaminen tai oman teeman kehittäminen. Valmiin teeman muokkaaminen vaatii toisen kirjoittamaan koodiin tutustumista, mutta kehittäjä voi selvitä vähemmällä työmäärällä, koska pohja on jo olemassa. Riskeinä ovat, että joudutaan kirjoittamaan valtava määrä koodia uudelleen tai ettei teema vain yksinkertaisesti taivu haluttuun tarkoitukseen. Oman teeman kehittäminen on taas työmäärältään suuri, mutta siinä saadaan juuri sitä, mitä halutaan.

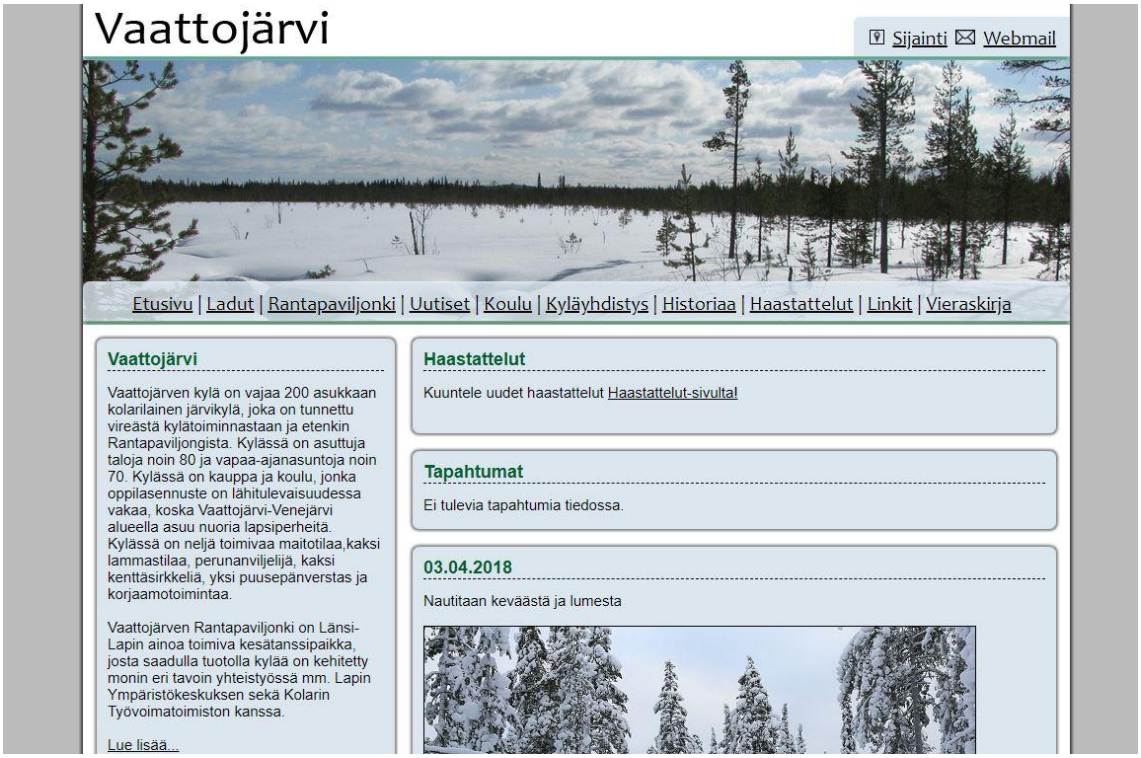
2 LÄHTÖKOHDAT JA TAVOITTEET

Tämän opinnäytetyön tavoitteena on toteuttaa uudet verkkosivut Vaattojärven kyläyhdistykselle. Kehitystehtävänä on toteuttaa uusi WordPress-teema käyttäen Bootstrap-kirjastoa, jolla ulkoasu saadaan päivitettyä noudattamaan nykyaikaista responsiivisuutta. Samalla saadaan myös käyttöön WordPressin helppokäyttöinen hallintapaneeli sivuston ylläpitoa ja päivitystä varten.

Vaattojärvi on noin 200 asukkaan kylä Kolarin kunnassa. Kylä tunnetaan erityisesti Länsi-Lapin ainoasta aktiivisesta kesätanssipaikasta, rantapaviljongista, jossa järjestetään tansseja aktiivisesti joka kesä. (Vaattojärvi 2018, viitattu 1.4.2018.)

2.1 Lähtötilanne

Vaattojärven nykyistä verkkosivustoa (kuva 1) ylläpidetään aktiivisesti lisäämällä sivustolle uutisartikkeleita. Lisäksi sivustolta löytyvät kylää koskevat tapahtumat sekä tanssitapahtumat. Kävijöiden määrää on mahdoton tarkemmin arvioida, sillä sivustolla ei ole kävijälaskuria. Kuitenkin vieraskirjan viestien määrästä voidaan päätellä, että sivustolla on aktiivinen käyttäjäkunta. Arvioisin, että käyttäjät ovat pääasiassa paikallisia, mutta myös ulkopaikkakuntalaisia kiinnostavat kesäisin järjestettävät tanssitapahtumat.



Kuva 1. Vaattojärven nykyisen sivuston ulkoasu

Sivustolla on oma yksinkertainen hallintapaneeli, jonka kautta voi muokata useiden sivujen tekstejä, luoda, muokata ja poistaa tapahtumia sekä uutisartikkeleja ja ylläpitää vieraskirjaa. Sen kautta ei kuitenkaan voi tehdä esimerkiksi haastattelut-sivuun, kuvabannereihin tai koulusivun kuvagalleriaan muutoksia, eli osa sivuston muutoksista täytyy vielä tehdä suoraan koodiin. Sivuston ylläpidon kannalta tämä on ongelma, sillä suurin osa ylläpitäjistä ei hallitse web-ohjelmointia.

Ulkoasu on myös jo vanhentunut. Sivusto on uudistettu viimeksi vuonna 2012 ja se näkyy. Sivuston ulkoasu on keskitetty, ja sen leveys on määrätty pikselintarkasti. Ulkoasu ei ole responsiivinen, joten sen käyttäminen pienemmillä näytöillä, kuten mobiililaitteilla, on hankalaa. Väriteemana on käytetty vaaleansinistä ja vihreää kuvastamaan paikallista luontoa, eli järveä ja metsää

2.2 Kehitystehtävät

Uusi sivusto tulee sisältämään seuraavat sivut: Etusivu, josta löytyvät uusin uutisartikkeli, tiedot tulevista tapahtumista ja tansseista, viimeisimmät haastattelut ja vieraskirjamerkinnät, lyhyen esittelyn ja kylän sijainnin. Vanhemmat uutisartikkelit ja haastattelut löytyvät omilta sivuiltaan. Rantapaviljonki-sivu esittelee kylän lavatanssipaikan sekä listaa menneet ja tulevat tanssitapahtumat.

Kyläyhdistyksen sivu sisältää nykyisen toimikunnan tiedot. Vaattojärven ja kyläyhdistyksen historiaan voi tutustua omalla historiasivulla. Sivulle toteutetaan myös vieraskirja, johon kävijät voivat kirjoittaa kuulumisiaan ja ilmoittaa muista järjestettävistä tapahtumista.

Kehitettävän sivuston vaatimuksina ovat nykyaikainen responsiivinen ulkoasu ja laajemmat ylläpito-ominaisuudet. Sivuston ylläpitämisen täytyy olla helppoa, ja sen tulee olla toteutettavissa ilman web-ohjelmoinnin osaamista, sillä kylätoimikunta ylläpitää sivustoa. Hallintapaneelin kautta tulee voida muuttaa sivuston kuvia, sivujen sisältöä, päivittää uutisartikkeleita sekä tapahtumia ja hallita vieraskirjaa. Vieraskirjassa täytyy myös olla esto roskapostiroboteille. Lisäksi hallintapaneelissa tulee nähdä vierailijoiden määrä sekä haastatteluiden kuuntelukerrat. Ulkoasun on tarkoitus olla näyttävä, mutta silti selkeä rakenteeltaan. Sivustoa käyttää myös moni vanhempi henkilö, joten selkeys on tärkeää. Varsinaiselle logolle ei ole tarvetta, joten sivuston otsikko tulee vain tekstinä. Väriteemana käytetään valkoista, mustaa ja harmaata, jotka ovat todella neutraaleja värejä. Nämä värit valittiin, koska ne sopivat muiden värien kanssa hyvin yhteen, joten ylätunnisteen taustakuvaksi voidaan valita mikä tahansa kuva.

Sivusto toimii Domainkeskuksen palvelimella, jota käytetään myös uuden sivuston kanssa. Uusi sivusto kuitenkin kehitetään erillisellä palvelimella, ja vasta sivuston valmistuttua korvataan nykyinen olemassa oleva sivusto uudella. Edellisen sivuston tulevat tapahtumat ja tanssitapahtumat siirretään uuteen sivustoon. Toimeksiantaja toimittaa kuvat uutta ulkoasua varten. Sisältö löytyy jo nykyiseltä sivustolta. Toimeksiantaja antaa viikoittain palautetta sivuston ulkoasusta ja käytettävyydestä puhelimitse.

Oppimisen kannalta tavoitteena on syventää osaamista WordPressin ja teemojen kehittämisessä. Samalla tutustutaan paremmin Bootstrap-kirjaston uuden julkaisun muutoksiin, sillä tämän vuoden alkupuolella Bootstrapista on julkaistu neljäs versio, joka tuo mukanaan paljon uudistuksia.

2.3 Työkalut

Laajemmat ja yksinkertaiset ylläpito-ominaisuudet saadaan kehittämällä sivusto sisällönhallintajärjestelmää käyttämällä. W3 Techs -sivuston (2018, viitattu 1.4.2018) tilastojen mukaan suurella osalla verkkosivustoista (49,5 %) ei ole mitään sisällönhallintajärjestelmää käytössä, mutta niiden

suosio on kasvanut viime vuosien aikana. WordPress on kaikista sisällönhallintajärjestelmistä suosituin, sillä sen markkinaosuus kaikista verkkosivustoista on tällä hetkellä 30,4%. Muita suosittuja vaihtoehtoja ovat Joomla (3,1 %) sekä Drupal (2,2 %). Kaikista sivustoista, jotka käyttävät sisällönhallintajärjestelmää, WordPressin osuus on 60,1 %. WordPressin markkinaosuus on kasvanut seitsemässä vuodessa 13,1 %:sta 30 %:iin (Soltano 2018, viitattu 1.4.2018). On todennäköistä, että WordPressin suosio tulee vain kasvamaan tulevaisuudessa sen jo vahvistuneen markkina-aseman vuoksi.

Tässä työssä käytän WordPress-sisällönhallintajärjestelmää sen vahvasti kasvavan suosion vuoksi. Olen lisäksi käyttänyt WordPressiä muissakin projekteissani ja todennut sen hyväksi sisällönhallintajärjestelmäksi niin kehittäjille kuin käyttäjillekin.

Ulkoasun responsiivisuus toteutetaan käyttämällä Bootstrap 4.1 -kirjastoa. Projektista julkaistiin 2018 alkupuolella neljäs vakaa versio, jonka suurimmat muutokset ovat: Flexbox on nyt oletuksena käytössä (Bootstrap 2018b, viitattu 10.4.2018). Flexbox on kolmanteen CSS-tasoon lisätty ulkoasumoduuli, joka helpottaa joustavien responsiivisten ulkoasurakenteiden kehitystä (W3Schools 2018a, viitattu 4.4.2018; W3C 2017, viitattu 24.5.2018). Glyphicons-fonttia ei enää toimiteta Bootstrapin mukana, mutta sen voi ladata erilliseltä sivulta, mikäli sitä tarvitsee. Muita vaihtoehtoisia ikonipaketteja ovat Octicons ja Font Awesome. Panel-, thumbnail- ja well-komponentit on poistettu kokonaan ja tilalle on tullut card-komponentti. Muita pienempiä muutoksia ovat muun muassa fonttikoon muuttaminen suuremmaksi 16px:iin, rem-mittayksikön ottaminen käyttöön, useiden hyötyluokkien lisääminen marginaalin (margin) ja täytteen (padding) helpompaan asetteluun sekä se, ettei enää ole tukea vanhemmille Internet Explorer 8 -, 9- ja iOS 6 -selaimille. (Bootstrap 2018b, viitattu 10.4.2018.)

3 TEEMAN KEHITYS

Teeman kehityksessä pääsee helpoimmalla, mikäli löytää sopivan valmiin teeman tai lähes sopivan teeman, josta luodaan sitten lapsiteema. Tätä lapsiteemaa voidaan muokata omiin tarpeisiin sopivaksi, eikä kaikkea tarvitse aloittaa alusta. Useita ilmaisiakin vaihtoehtoja on, mutta parhaimmista teemoista voi joutua maksamaan. Mikäli kyseessä ei ole suuri summa, tähän mahdollisuuteen kannattaa yleensä tarttua, sillä teeman kehittäminen on aikaa vievää ja vaatii WordPressin teknisen puolen ymmärtämistä.

Aina ei ole olemassa valmista teemaa, joka tarjoaisi ne ominaisuudet tai ulkoasun, jota ollaan etsimässä. Mikään ei estä kehittämästä omaa teemaa, ja sen mahdollinen ilmainen levittäminen tai myyminen on helppoa eri palveluntarjoajien kautta. Oman teeman kehityksen etuja ovat uniikki ulkoasu, uuden oppiminen, mahdolliset myyntitulot ja helppo muokattavuus: Kun teeman kehittää itse, ei tarvitse tutustua toisen henkilön tekemään koodiin ja mahdolliseen dokumentaatioon.

WordPressin virallisella sivustolla on Codex- (https://codex.wordpress.org) ja Developer-osio (https://developer.wordpress.org), joista kehittäjät löytävät tietoa, teknistä tukiaineistoa ja oppaita WordPressin kehitykseen. Teemojen ulkoasu kehitetään HTML- ja CSS-koodia käyttäen, mutta ulkoasun integrointi WordPressiin toteutetaan PHP-koodilla ja WordPressiin on ohjelmoitu paljon valmiita funktioita, joten kaikkea ei tarvitse kirjoittaa alusta.

3.1 Teeman rakenne

WordPress-teeman kehitys vaatii vähintään kaksi pakollista tiedostoa: tyylitiedoston ”style.css” sekä sivupohjan ”index.php”. Kummatkin sijoitetaan teeman omaan alikansioon, joka tulee tehdä wp-content/themes/-hakemistoon. Teeman kansion nimi ei saa sisältää numeroita, sillä se estää teeman näkymisen käytettävien teemojen listalla. (WordPress 2018c, viitattu 11.4.2018.)

WordPressin virallisen dokumentaation (WordPress 2018a, viitattu 11.4.2018) mukaan WordPressin mukana toimitettavat Twenty-teemat ovat parhaimpia esimerkkejä hyvästä teeman tiedostojen organisoinnista. Esimerkiksi uusin Twenty Seventeen -teema (kuva 2) on lajitellut kaikki päätiedostot, kuten sivupohjat, virhesivun, ylä- ja alatunnisteen, sivupalkin ja ruudunkaappauskuvan teeman

kansion juureen. JavaScript-, CSS- ja kuvatiedostot löytyvät jokainen omista kansioistaan assets-kansion alta. Inc-kansiossa on kokoelma teemalle oleellisia funktioita, joita functions.php kutsuu. Sivupohjia on jaettu osiin template-parts-kansioon, jotta itse sivupohjat pysyvät helppolukuisina.

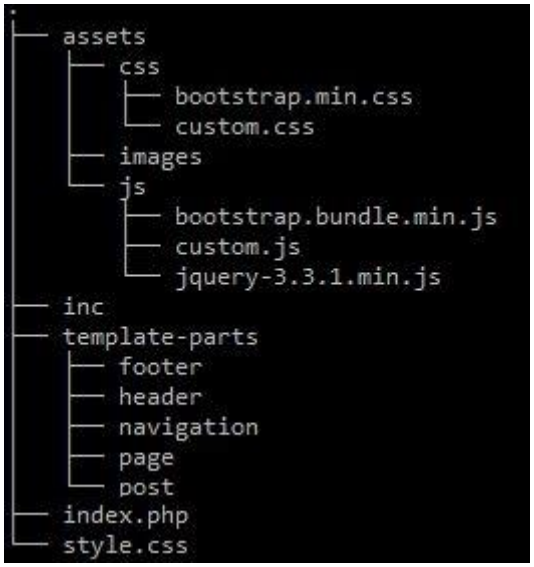
Kuva 2. Twenty Seventeen -teeman tiedostorakenne

Uuden teeman kehitys lähtee liikkeelle luomalla teemalle oma kansio ”vaattojarvi” wp-content/themes/-hakemistoon. Teeman omaan kansioon luodaan myös tiedostot ”index.php” ja ”style.css”, jotka mahdollistavat teeman käyttöönoton WordPressin hallintapaneelista. Käyttäen hyvää esimerkkiä Twenty Seventeen -teemasta (kuva 2), luodaan samanlainen tiedostorakenne (kuva 3). Assets-hakemistoon voidaan myös valmiiksi lisätä uudet tyyli- ja JavaScript-tiedostot, esimerkiksi custom.css ja custom.js, joita käytetään teeman tyylimäärityksiin ja toiminnallisuuksiin.

Bootstrap-kirjaston käyttämiseen tarvitaan Bootstrapin oma CSS-tyylitiedosto, oma JavaScript-kirjasto, uusin jQuery-kirjasto sekä popper.js-kirjasto. Popper.js-kirjasto on lisätty Bootstrapin neljänteen versioon pudotusvalikoiden, vihjelaatikoiden ja popover-komponentin näyttämiseen sekä asetteluun. (Bootstrap 2018c, viitattu 23.4.2018.) Bootstrap-jakelun mukana tulee JavaScript-paketti bootstrap.bundle.js, joka sisältää molemmat bootstrap.js- ja popper.js-tiedostot, mutta uusin jQuery-kirjasto täytyy edelleen ladata erikseen omalta viralliselta sivustolta (http://jquery.com/download) (Bootstrap 2018f, viitattu 1.5.2018). Bootstrapin käyttöön riittää myös jQuery-kirjaston slim-versio, josta on poistettu ajax- ja effects-moduulit. (Bootstrap 2018c, viitattu 23.4.2018; jQuery 2018, viitattu 1.5.2018).

Tarvittavat tiedostot jquery-3.3.1.min.js, bootstrap.bundle.min.js ja bootstrap.min.css sijoitetaan kansiorakenteen mukaisesti assets/js- ja assets/CSS-kansioihin. Kyseisistä tiedostoista käytetään

pakattuja versioita (minified), joista on poistettu muun muassa kommentointi ja formatointi, jotta niiden latausaika olisi mahdollisimman pieni.



Kuva 3. Uuden teeman alustava tiedostorakenne

3.2 Teeman määritys

Tyylitiedostoa style.css voisi ajatella teeman tärkeimpänä tiedostona, sillä teemaa ei voi valita käyttöön ilman sitä. Virallisen dokumentaation (WordPress 2018c, viitattu 11.4.2018) mukaan tyylimääritysten lisäksi style.css välittää teeman tiedot WordPressille kommenttien muodossa.

Esimerkiksi Twenty Seventeen -teeman style.css (kuva 4) määrittää ensimmäisenä teeman nimen. Lisäksi teeman tietoihin voidaan lisätä teeman osoite, tekijän nimi, tekijän verkkosivu, kuvaus teemasta, versionumero, teeman lisenssi, lisenssin osoite ja teemaa kuvaavat avainsanat. Avainsanat eivät kuitenkaan voi olla mitä tahansa, vaan avainsanat ja niiden vaatimukset ovat tarkasti määritetty virallisessa dokumentaatiossa (WordPress 2018g, viitattu 2.5.2018). Hyvän tavan mukaisesti teemasta voidaan tehdä muille kielille käännettävä, vaikka kehittäjä ei sitä itse kääntäisikään. Jotta käännöstiedostot toimivat oikein, style.css-tiedostoon täytyy määrittää ”text domain”, jolla merkitään teemalle kuuluvat käännöstekstit, ja sen täytyy olla sama kuin teeman kansion nimi (WordPress 2018h, viitattu 2.5.2018).

```
/*
Theme Name: Twenty Seventeen
Theme URI: https://wordpress.org/themes/twentyseventeen/
Author: the WordPress team
Author URI: https://wordpress.org/
Description: Twenty Seventeen brings your site to life with header video and immersi
Version: 1.4
License: GNU General Public License v2 or later
License URI: http://www.gnu.org/licenses/gpl-2.0.html
Text Domain: twentyseventeen
Tags: one-column, two-columns, right-sidebar, flexible-header, accessibility-ready,

This theme, like WordPress, is licensed under the GPL.
Use it to make something cool, have fun, and share what you've learned with others.
*/
```

Kuva 4. Twenty Seventeen -teeman style.css:n sisältö

Uuden teeman tyhjään style.css-tiedostoon täydennetään Twenty Seventeen -teeman style.css-tiedoston esimerkin mukaisesti teeman nimi, tekijän nimi, lyhyt kuvaus ja versionumero voi olla esimerkiksi 0.1. Teeman täytyy hyvän tavan mukaisesti tukea käännöksiä, joten määritetään myös ”Text Domain”. Avainsanoja ja lisenssiä ei tarvita, koska teema tulee vain omaan käyttöön. Nyt teema voidaan ottaa käyttöön WordPressin hallintapaneelista.

3.3 Teeman toiminnot

Tiedostoa functions.php voidaan käyttää lisäämään teemaan ominaisuuksia ja toiminnallisuutta. WordPress suorittaa aktiivisen teeman kansion juuressa olevan functions.php-tiedoston automaattisesti, joten sen avulla voidaan kutsua WordPressin sisäisiä funktioita ja määrittää omia. (WordPress 2018j, viitattu 2.5.2018.) Esimerkiksi Twenty Seventeen -teeman functions.php sisältää muun muassa tyyli-, JavaScript- ja käännöstiedostojen latauksen, teeman käytön rajoittamisen uudempaan kuin WordPressin 4.7-alphaversioon, teeman tukemien ominaisuuksien, valikoiden ja kuvien koon määrityksen.

Oikea tapa tuoda omat tyylitiedostot WordPress-teemaan on käyttää wp_enqueue_style()-funktiota (kuva 5). Funktio tarvitsee uniikin nimen tyylitiedostolle ja lisäksi voidaan syöttää lisäparametreina tyylitiedoston sijainti, tyylitiedostojen nimet, joista tämä tyylitiedosto on riippuvainen, versionumero ja media. Mikäli versionumeroa ei syötetä, WordPress antaa tiedostolle WordPressin versionumeron. Medialla voidaan määrittää, mitä mediaa tyylitiedosto koskee, kuten esimerkiksi tulostusnäkymää ”print”. (WordPress 2018d, viitattu 12.4.2018.)

JavaScript-tiedostojen tuominen WordPress-teemaan tapahtuu lähes samanlaisella funktiolla wp_enqueue_script() (kuva 5). Kyseinen funktio tarvitsee uniikin nimen ja lisäksi voidaan syöttää lisäparametreina tiedoston sijainti, tiedostojen nimet, joista tämä tiedosto on riippuvainen, versionumero ja ladataanko tiedosto ennen </body>-tagia <head>-tagin sijaan. Mikäli versionumeroa ei syötetä, WordPress antaa tiedostolle WordPressin versionumeron. (WordPress 2018i, viitattu 2.5.2018.)

```
function twentyseventeen_scripts() {
    ...

    // Theme stylesheet.
    wp_enqueue_style(
        'twentyseventeen-style',
        get_stylesheet_uri() );

    ...

    wp_enqueue_script(
        'twentyseventeen-global',
        get_theme_file_uri( '/assets/js/global.js' ),
        array( 'jquery' ), '1.0', true );

    ...
}
add_action( 'wp_enqueue_scripts', 'twentyseventeen_scripts' );
```

Kuva 5. Twenty Seventeen -teeman skriptien latausfunktio

Tiedosto functions.php luodaan Vaattojärven teeman kansion juureen, johon kirjoitetaan aluksi kaksi funktiota: vaattojarvi_scripts() ja vaattojarvi_setup(), joilla ladataan CSS-, JavaScript- ja käännöstiedostot (kuva 6). Funktio vaattojarvi_scripts() sisältää kuusi funktiota, joilla ladataan Bootstrapin CSS-tiedosto, teeman oma CSS-tiedosto, WordPressin mukana tuleva jQuery 1.12.4, jQuery 3.3.1, Bootstrapin JavaScript-paketti ja teeman oma JavaScript-tiedosto. Teeman omalle tyylitiedostolle on määritetty riippuvuus Bootstrapin tyylitiedostosta, joka täytyy ottaa käyttöön ensin. Bootstrapin JavaScript-paketille ja teeman omalle JavaScript-tiedostolle on määritetty riippuvuus uudempaan jQuery-jakeluun, joka täytyy ottaa käyttöön ennen kyseessä olevien tiedostojen ajamista. Vanhan jQueryn voi poistaa käytöstä wp_deregister_script()-funktiolla, mutta suurin osa WordPressin lisäosista käyttää edelleen vanhaa jQuery-jakelua, joten sen poistaminen käytöstä tulee vaihtoehdoksi vain, mikäli kehittäjä tietää tarkkaan, mitä lisäosia teeman loppukäyttäjä tarvitsee. Näitä funktioita kutsutaan toimintakutsuilla wp_enqueue_scripts ja after_setup_theme, joista ensimmäinen on tarkoitettu lataamaan tyyli- ja JavaScript-tiedostot ja jälkimmäinen kutsuu funktiota aina jokaisen sivun latauksen aikana.

```php
function vaattojarvi_scripts() {

wp_enqueue_style( 'bootstrap41css',
    get_stylesheet_directory_uri() . '/assets/css/bootstrap.min.css',
    array(), '4.1' );
wp_enqueue_style(
    'customcss',
    get_stylesheet_directory_uri() . '/assets/css/custom.css',
    array('bootstrap41css') );

wp_enqueue_script('jquery');
/* wp_deregister_script('jquery'); */

wp_register_script(
    'jquery331',
    get_template_directory_uri() . '/assets/js/jquery-3.3.1.min.js',
    array(), '3.3.1', true );

wp_enqueue_script(
    'bootstrap41js',
    get_template_directory_uri() . '/assets/js/bootstrap.bundle.min.js',
    array('jquery331'), '4.1', true );

wp_enqueue_script(
    'customjs',
    get_template_directory_uri() . '/assets/js/custom.js',
    array('jquery331'), '1.0', true );

}
add_action( 'wp_enqueue_scripts', 'vaattojarvi_scripts' );

function vaattojarvi_setup() {

    load_theme_textdomain( 'vaattojarvi' );

}
add_action( 'after_setup_theme', 'vaattojarvi_setup' );
```

Kuva 6. Vaattojärven teeman functions.php

4 ULKOASU

Ulkoasu muodostuu ylä- ja alatunnisteesta sekä sivupohjista. Ylätunnisteeksi kutsutaan sivuston yläosaa, johon tässä tapauksessa kuuluvat head-, header- ja nav-elementit. Ylätunnisteessa määritetään sivuston kieli, merkistö, kuvaus, metatagit, logo, logon taustakuva ja navigaatiovalikko. Alatunnisteeksi kutsutaan sivuston alaosaa, johon tässä tapauksessa kuuluu vain footer-elementti sekä body- ja html-tagin sulkeminen. Footer-elementtiin sijoitetaan sivuston nimi. Itse sivupohjiin sijoitetaan kaikki, joka halutaan näkyvän ylä- ja alatunnisteen välissä, kuten sivun sisältö. Sivupohjat voidaan kustomoida jokaiselle sisällölle erikseen, joten halutessa voidaan käyttää esimerkiksi etusivulle, sivuille ja artikkeleille erilaisia sivupohjia.

4.1 Ylätunniste

Kaikki koodi, joka tulee ennen sivuston sisältöä, voidaan tallentaa omaan tiedostoon, joka voidaan tuoda aina uuteen sivupohjaan funktion avulla, eli sivuston yläosan koodia ei tarvitse kirjoittaa aina uudestaan. WordPressin Codexin mukaan (WordPress 2018k, viitattu 3.5.2018) ylätunnistetiedosto voidaan tuoda funktiolla get_header(), jolle voidaan määrittää lisäparametrinä nimi. Tiedosto täytyy nimetä joko header.php:ksi tai header-nimi.php:ksi, sillä ilman header-termiä, funktio tuo WordPressin mukana toimitettavan ylätunnistetiedoston (wp-includes/theme-compat/header.php).

Kun header.php-tiedostoa kirjoitetaan, avatun html-tagin täytyy sisältää language_attributes()-funktio, johon WordPress täydentää käytössä olevan kielen ja lukusuunnan (WordPress 2018e, viitattu 12.4.2018). Html-tagiin ei siis pidä itse määrittää mitään, vaan antaa funktion hoitaa määritys automaattisesti WordPressin asetuksissa asetetun kielen mukaan. Lisäksi Bootstrapin käyttöönotto vaatii HTML5-doctypen sekä responsiivisen viewport-metatagin määrityksen toimiakseen oikein (Bootstrap 2018c, viitattu 23.4.2018).

Jotta selain osaa näyttää HTML-sivun oikein, sivulle täytyy määrittää, mitä merkistöä (charset) sivu käyttää (W3Schools 2018c, viitattu 16.4.2018). Meta-elementti, joka sisältää WordPressin oman bloginfo('charset')-funktion, tulee sijoittaa ensimmäiseksi avatun head-tagin alle. (WordPress

2018c, viitattu 11.4.2018). Bloginfo('description')-funktiolla voidaan hakea sivuston kuvaus (description) toista meta-elementtiä varten, joka määritetään WordPressin hallintapaneelista (WordPress 2018f, viitattu 16.4.2018).

Sivun title-tagi, eli otsikko, voidaan hakea WordPressin funktiolla wp_title(), mutta suositeltu tapa on määrittää functions.php-tiedostoon teemalle tuki ”title-tagille” funktiolla add_theme_support('title-tag'), jolloin WordPress automaattisesti asettaa title-tagin head-tagin sisään. Kyseinen funktio käyttää otsikossa muotoa ”sivun nimi - sivuston nimi”, kuten esimerkiksi ”Historia - Vaattojärvi”. Tätä muotoa suositellaan käytettäväksi hakukoneoptimoinnissa. (WordPress 2018l, viitattu 3.5.2018; Obénland, K. 2014, viitattu 3.5.2018.)

Ennen head-tagin sulkemista täytyy kutsua wp_head()-toimintakutsua, jota lisäosat käyttävät lisätäkseen omia skriptejä, tyylimäärityksiä ja muita toiminnallisuuksia (WordPress 2018c, viitattu 11.4.2018). Head-tagin sulkemisen jälkeen tulee määrittää avatulle body-tagille body_class()-lisämääritys, jota esimerkiksi lapsiteemat ja lisäosat voivat käyttää hyväkseen muokatakseen teeman ulkoasua (WordPress 2018m, viitattu 4.5.2018).

Vaattojärven teemalle luodaan tiedosto header.php (kuva 7) teeman kansion juureen, johon määritetään ensimmäisenä doctypeksi html ja html-tagiin asetetaan WordPressin language_attributes()-funktio. Head-tagin sisään määritetään WordPressin funktioiden avulla meta-tagit charset, viewpoint ja description. Viewport-metatagin koodi on suoraan kopioitu Bootstrapin ”Starter template” -mallista, joka takaa Bootstrapin toimivuuden. Ennen head-tagin sulkemista kutsutaan wp_head()-funktiota ja head-tagin sulkemisen jälkeen määritetään body-tagille body_class()-funktio. Lisäksi functions.php-tiedostoon lisätään vaattojarvi_setup()-funktion alle tuki automaattiselle title-tagin otsikoinnille add_theme_support('title-tag')-ominaisuudella.

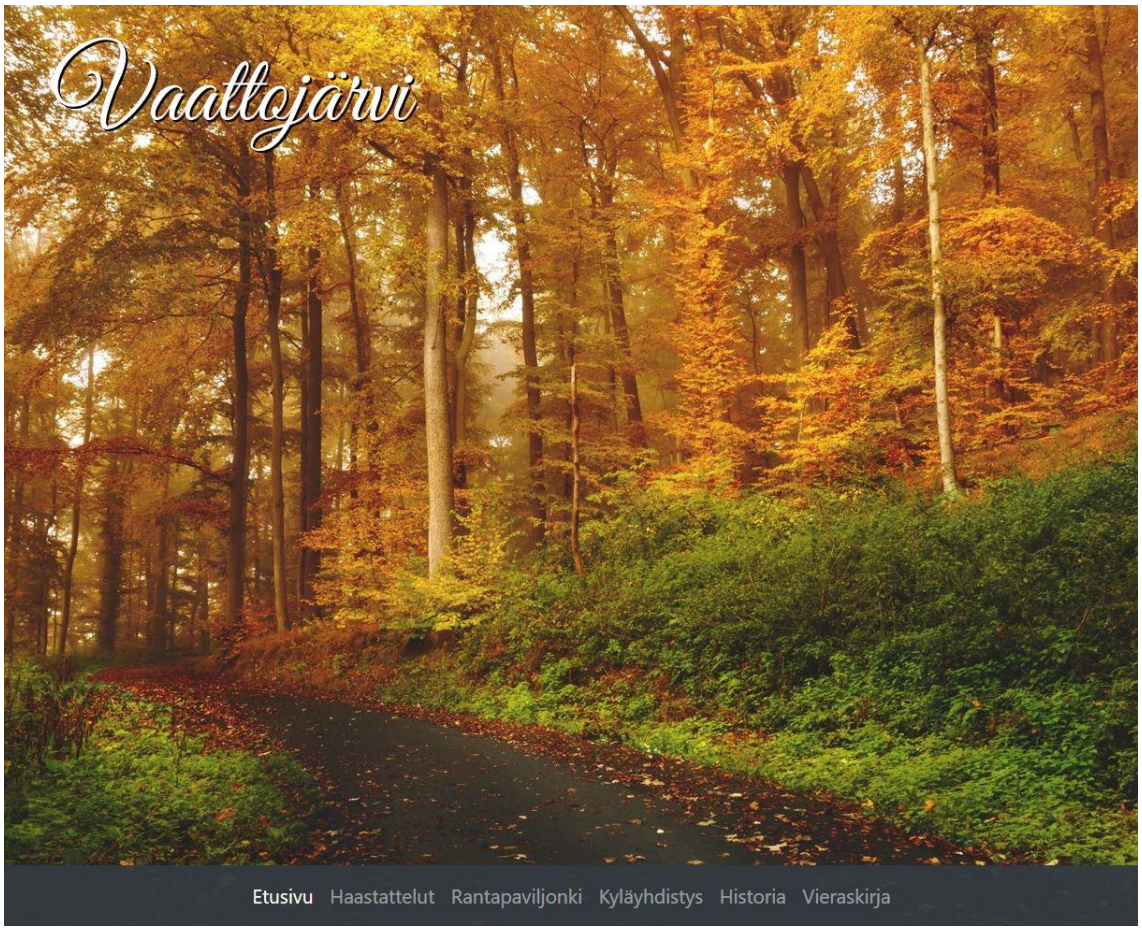
```
<!doctype html>
<html <?php language_attributes(); ?>>
<head>

    <meta charset="<?php bloginfo( 'charset' ); ?>">
    <meta name="viewport" content="width=device-width, initial-scale=1,
    shrink-to-fit=no">
    <meta description="<?php bloginfo( 'description' ); ?>">

    <?php wp_head(); ?>
</head>
<body <?php body_class(); ?>>
```

Kuva 7. Doctypen, head-, meta- ja body-tagien määritys header.php-tiedostoon

Vaattojärven teeman ylätunnisteessa (kuva 8) on koko näytön peittävän taustakuva, sivuston nimi sekä navigaatiovalikko. Taustakuva asetetaan header-elementtiin, joka rullatessa alaspäin pienenee animaatiolla 400 pikseliin työpöydällä ja mobiililaitteilla 250 pikseliin. Muilla kuin etusivulla, ylätunniste on aina valmiiksi pienennetty. Sivuston nimelle tehdään header-elementin sisälle oma elementtinsä, jonka asettelu hoidetaan Bootstrapin luokilla. Navigaatio tulee header-elementin alle omaksi pääelementiksi, mutta se siirretään tyylimäärityksillä taustakuvan päälle. Vaattojärvi-otsikon tausta on täysin läpinäkyvä ja navigaatiovalikon tausta on 5 % läpinäkyvä.



Kuva 8. Vaattojärven etusivun ylätunniste

Bootstrapin Grid-systeemi on uudistettu käyttämään Flexboxia, joten Bootstrapista löytyy parempi tuki sarakkeiden pysty- ja vaakasuoraan asetteluun Bootstrapin omien luokkien kautta. Lisäksi Grid-systeemiin on lisätty uusi taso, eli tasoja ovat nyt: xs, sm, md, lg ja xl (kuva 9). Pienimmän tason, eli xs-tason käyttöön, ei enää määritetä breakpointia. (Bootstrap 2018b, viitattu 9.4.2018.)

	Extra small <576px	Small ≥576px	Medium ≥768px	Large ≥992px	Extra large ≥1200px
Max container width	None (auto)	540px	720px	960px	1140px
Class prefix	.col-	.col-sm-	.col-md-	.col-lg-	.col-xl-

Kuva 9. Bootstrap 4:n uudet breakpointit (Bootstrap 2018d, viitattu 23.4.2018)

Header-elementille (kuva 10) määritetään luokka ”container-fluid”, joka on täysileveä säiliö row- ja col-luokille. Elementin ympärille ei haluta ylimääräistä täytettä, joten tyylimäärityksiin asetetaan container-fluid-luokalle ”padding: 0”. Tämän säiliön sisälle luodaan row- ja col-luokilla määritetyt

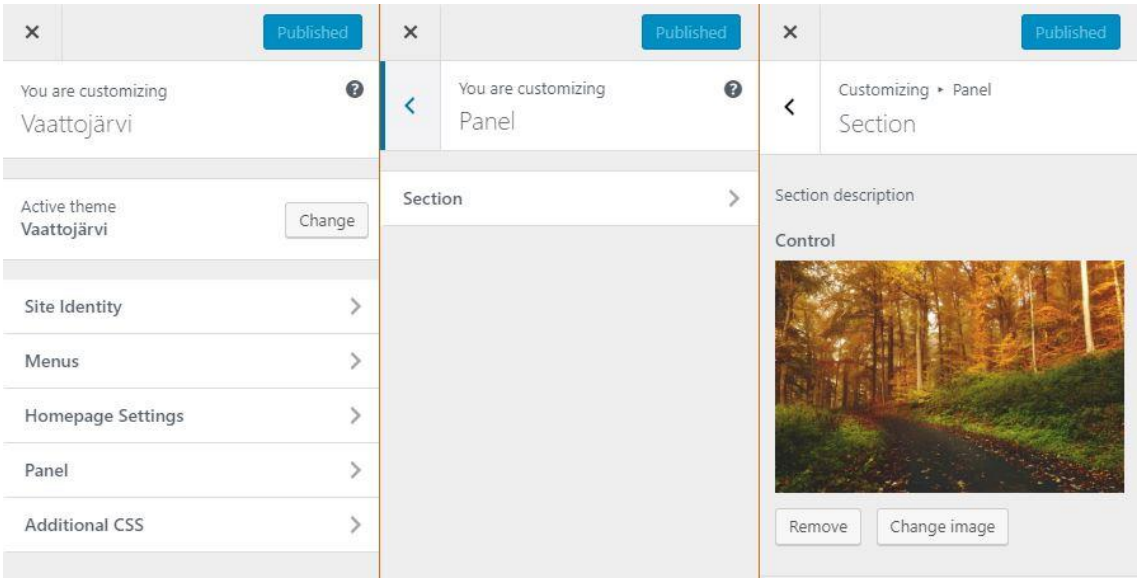
div-elementit, joiden sisään asetetaan sivuston nimi. Annetaan col-luokan elementille id ”logo”, jonka avulla määritetään tyylitiedostoon tekstin väri, tekstin koko, täyte, tekstin suunta, eri fontti sekä taustan väri. Oletuksena col-luokka varaa elementille kaiken tilan vasemmalta ja oikealta puolen elementtiä, joten muutetaan col-luokka col-auto-luokaksi, joka varaa vain sisällön tarvitseman tilan elementille. WordPressiin määritetty sivuston nimi voidaan hakea funktiolla bloginfo(’name’) (WordPress 2018f, viitattu 16.4.2018).

```
<header class="container-fluid">
    <div class="row">
        <div class="col-auto offset-xl-2" id="logo">
            <?php bloginfo( 'name' ); ?>
        </div>
    </div>
</header>
```

Kuva 10. Header-elementin määritys

4.1.1 Mukauta-valikko

Ylätunnisteen taustakuvan halutaan olevan vaihdettavissa, joten sille täytyy tehdä oma asetus WordPressin mukauta-valikkoon. WordPressin dokumentaation mukaan teemoille ja lisäosille voidaan luoda mukauta-valikkoon omia paneeleja (panel), lohkoja (section), asetuksia (settings) sekä ohjaimia (controls). Kuten kuvasta 11 nähdään, paneeleilla voidaan luoda kokonaan oma asetusvalikko, jonka sisään voidaan luoda lohkoja. Lohkojen sisään sijoitetaan lohkoon liittyvät ohjaimet. Virallinen dokumentaatio ei kuitenkaan suosittele omien paneelien tekemistä, koska ne ovat suunniteltu suuremmille kokonaisuuksille, kuten vimpaimille. (WordPress 2018n, viitattu 5.5.2018.)



Kuva 11. WordPressin panel, section ja control

Mukauta-valikkoa voidaan muokata ”Customize Manager” -objektilla, joka saadaan käyttöön customize_register-toimintakutsulla. Kyseisen objektin avulla voidaan lisätä, muokata ja poistaa paneeleja, lohkoja, ohjaimia ja asetuksia. Asetukset käsittelevät objektin esikatselun, tallennuksen ja tarkistuksen, mutta asetukset tarvitsevat ohjainobjektin hallinnoimaan niitä. (WordPress 2018n, viitattu 5.5.2018.) Asetukset voi ajatella objekteina, joihin viitataan sivun koodissa ja ohjainobjektit taas mahdollistavat asetusobjektin muokkauksen WordPressin mukauta-valikossa.

Paneeli-, lohko- ja ohjainobjektille täytyy määrittää vähintään id ja nimike (label). Mikäli paneeleja on useampi, lohko-objektille täytyy määrittää, mihin paneeliin kyseinen lohko kuuluu. Ohjainobjektille taas täytyy määrittää, mihin lohkoon ja mille asetukselle kyseinen ohjain kuuluu. Asetusobjekti tarvitsee vähintään id:n, jolla objekti tunnistetaan ja erotellaan muista objekteista. Asetusobjektin sisältö voidaan hakea funktiolla get_theme_mod, jolle täytyy määrittää haettavan asetusobjektin nimi (id) ja valinnaisesti oletusarvo. (WordPress 2018n, viitattu 5.5.2018.)

Muiden tiedostojen tuominen onnistuu PHP:n funktiolla require tai include. Mikäli tiedoston tuonti ei onnistu, require-funktio tulostaa ”fatal errorin” ja ei lataa sivua ollenkaan, kun taas include-funktio tulostaa vain varoituksen, mutta lataa silti sivun loppuun. Twenty Seventeen -teemassa mukauta-valikon muokkausfunktio ja toimintakutsu on kirjoitettu erilliseen tiedostoon customizer.php, joka on sjoitettu inc-kansioon, josta functions.php kutsuu kyseistä tiedostoa funktiolla require.

Kuten Twenty Seventeen -teemassa, luodaan uusi funktio ”vaattojarvi_customize_register” customizer.php-tiedostoon, joka sijoitetaan inc-kansioon. Tätä funktiota kutsutaan toimintakutsulla add_action('customize_register','vaattojarvi_customize_register'), jonka avulla ”Customize Manager” -objekti saadaan käyttöön. Funktion sisään kirjoitetaan kaksi metodikutsua add_setting ja add_control. Kuten kuvasta 12 nähdään, add_setting-metodikutsuun on määritetty id:ksi ”vaattojarvi_header_bg_setting” ja oletusparametriksi oletustaustakuvan tiedostopolku. Toiseen metodikutsuun add_control on määritetty id:ksi ”vaattojarvi_header_bg_control”, nimikkeeksi ”Ylätunnisteen taustakuva”, lohkoksi ”title_tagline” ja asetusobjektin id, jota ohjainobjekti hallinnoi. Lohkomääritys ”title_tagline” sijoittaa ohjainobjektin ”Sivuston identiteetti” -lohkon alle.

```
function vaattojarvi_customize_register ($wp_customize) {
    $wp_customize->add_setting( 'vaattojarvi_header_bg_setting', array(
        'default' => get_template_directory_uri() . '/assets/images/header_default_bg.jpg'
        )
    );
    $wp_customize->add_control(
        new WP_Customize_Image_Control(
            $wp_customize,'vaattojarvi_header_bg_control',array(
                'label' => 'Ylätunnisteen taustakuva',
                'section' => 'title_tagline',
                'settings' => 'vaattojarvi_header_bg_setting',
            )
        )
    );
}
add_action('customize_register','vaattojarvi_customize_register');
```

Kuva 12. Vaattojärven teeman customizer.php-tiedoston customize_register-toimintakutsu

Tällä hetkellä taustakuvan voi vaihtaa WordPressin mukauta-valikon kautta, mutta kuva täytyy vielä lisätä header-elementtiin, jotta se näkyy sivulla. Asetusobjektin sisältö saadaan haettua funktiolla get_theme_mod('vaattojarvi_header_bg_setting'), joka sijoitetaan muuttujan kautta header-elementin style-määritykseen (kuva 13). Lisäksi WordPressin sisäänrakennetulla funktiolla is_front_page() voidaan tarkistaa, onko sivulla auki etusivu vai jokin muu sivu, jonka perusteella näytetään joko suuri tai pienennetty taustakuva (WordPress 2018o, viitattu 6.5.2018).

```
<?php $theme_header_bg = get_theme_mod('vaattojarvi_header_bg_setting'); ?>
<?php if ( is_front_page() ) : ?>
<header
    id="frontpage"
    class="container-fluid"
    style="background-image:url('<?php echo $theme_header_bg ?>');">
<?php else : ?>
<header
    class="container-fluid"
    style="background-image:url('<?php echo $theme_header_bg ?>');">
<?php endif; ?>
    <div class="row">
        <div class="col-auto offset-xl-2" id="logo">
            <?php bloginfo( 'name' ); ?>
        </div>
    </div>
</header>
```

Kuva 13. Funktiolla get_theme_mod haetaan asetusobjektin sisältö

4.1.2 Navigaatiovalikko

Bootstrapin navbar-komponentti on kirjoitettu kokonaan uusiksi käyttämään flexboxia tarjotakseen paremman tuen komponentin asettelulle, responsiivisuudelle ja muokattavuudelle. Komponentille täytyy määrittää breakpoint, jolla määritetään, missä kohtaa komponentti siirtyy mobiilinäkymään, eli listaa linkit allekkain. Uusi navbar vaatii myös tekstin ja taustan värin määrittämisen: Tekstille on valittavissa luokat ”navbar-light” tai ”navbar-dark”, joka määrittää tekstin värin tummaksi tai vaaleaksi. Taustalle on valittavissa ”bg-light” tai ”bg-dark”, joka määrittää taustan vaaleaksi tai tummaksi. Taustavärin voi myös asettaa haluamakseen tyylimäärityksellä ”background-color”, jolloin taustan luokkaa ei tarvitse asettaa. (Bootstrap 2018b, viitattu 10.4.2018.) Navbar-komponentti tarvitsee toimiakseen a-elementeille luokan ”nav-link”, jokainen a-elementti täytyy sijoittaa erikseen oman li-elementtin sisälle, joka tarvitsee luokan ”nav-item”, li-elementit täytyy sijoittaa ul-elementin sisälle, joka tarvitsee luokan ”navbar-nav” ja viimeiseksi ul-elementti täytyy sijoittaa div-elementin sisään, jonka luokat ovat ”collapse” ja ”navbar-collapse”. (Bootstrap 2018g, viitattu 12.5.2018)

WordPressin sisäänrakennettu navigaatiovalikko voidaan näyttää funktiolla wp_nav_menu(). Navigaatiovalikkoa voidaan muokata funktion lisäparametreillä, joilla voidaan asettaa muuan muassa id- ja class-määrityksiä navigaatiovalikon elementteihin, tekstiä ennen linkin a-elementin aloitusta ja sen jälkeen, kuinka monta hierarkiatasoa pudotusvalikoissa otetaan mukaan ja walker-luokka. (WordPress 2018p, viitattu 12.5.2018.) Myös navigaatiovalikon ul-elementin sisällä olevien li-elementtien luokkia voidaan muokata nav_menu_css_class-toimintakutsulla (WordPress 2018q, viitattu 12.5.2018).

WordPressin navigaatiovalikon rakenne on lähes samanlainen kuin Bootstrapin navbar-komponentin rakenne. Kaikki muut navbar-komponentin tarvitsemat luokat voitaisiin sijoittaa WordPressin navigaatiovalikkoon wp_nav_menu-funktiolla ja nav_menu_css_class-toimintakutsulla, paitsi a-elementin tarvitsema nav-link-luokka. Linkkielementin luokan määrittämiseen WordPress ei tarjoa ratkaisua, joten ainoa vaihtoehto on käyttää walker-luokkaa.

Walker-luokka on tarkoitettu esittämään puutietorakenteisia listoja ja objekteja, jotka sisältävät hierarkkista dataa. Puutietorakenteen omaavia WordPress-objekteja ovat esimerkiksi navigaatiovalikot ja sivukategoriat. Walker-luokka käy läpi jokaisen puutietorakenteen solmukohdan ja suorittaa jokaisen kohdalla funktion, joihin kehittäjä voi vaikuttaa lapsiluokan kautta. (WordPress 2018r, viitattu 14.5.2018.)

WP Bootstrap Navwalker on mukautettu walker-luokka WordPressiin, joka formatoi navigaatiovalikon uudelleen Bootstrap-kirjaston navbar-komponenttia varten. Kyseinen walker-luokka täytyy ladata suoraan Githubista, tuoda tiedosto WordPressiin PHP-ympäristön require_once-funktiolla ja ottaa käyttöön wp_nav_menu-funktion walker-lisäparametrillä. (WP Bootstrap Navwalker 2018, viitattu 14.5.2018.)

Teeman ylätunnistetiedoston header-elementin alle luodaan uusi elementti ”nav” (kuva 14), jonka luokiksi asetetaan ”navbar”, ”navbar-expand-lg” ja ”navbar-dark”. Navbar-luokka määrittää kyseessä olevan navbar-komponentti, lg-breakpoint muuttaa navigaatiolinkit mobiilinäkymään alle 960 pikselin leveydessä ja ”navbar-dark” määrittää tekstin värin vaaleaksi. Taustan väri määritetään tyylitiedostossa. Funktiolle wp_nav_menu on määritetty lisäparametrit depth, walker, container, menu_class ja fallback_cb, joilla määritetään pudotusvalikot ja ladattu walker-luokka käyttöön, container-elementti pois käytöstä, ul-elementille luokka ”navbar-nav” sekä ladatun walker-luokan dokumentaatiossa esitetty fallback-funktio. Mukautettu walker-luokka ladataan WordPressiin functions.php-tiedostossa funktiolla require_once.

```
<nav class="navbar navbar-expand-lg navbar-dark">
    <div class="container">
        <button
            aria-controls="navbarSupportedContent"
            aria-expanded="false"
            aria-label="Toggle navigation"
            class="navbar-toggler"
            data-target="#navbarSupportedContent"
            data-toggle="collapse" type="button">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div
            class="collapse navbar-collapse justify-content-center"
            id="navbarSupportedContent">
            <?php
            wp_nav_menu( array(
            'depth'             => 2,
            'container'         => false,
            'menu_class'        => 'navbar-nav',
            'fallback_cb'       => 'wp_bootstrap_navwalker::fallback',
            'walker'            => new wp_bootstrap_navwalker())
            );
            ?>
        </div>
    </div>
</nav>
```

Kuva 14. Teeman navigaatiovalikko

4.2 Alatunniste

Kaikki koodi, joka tulee sivuston sisällön jälkeen, voidaan myös tallentaa omaan tiedostoon, joka voidaan tuoda jokaiseen sivupohjaan funktion avulla. Ennen body-tagin sulkemista täytyy määrittää wp_footer-funktio, jota muun muassa lisäosat, kuten esimerkiksi Google Analytics, käyttävät sijoittaakseen koodinsa ajettavaksi kaiken muun koodin jälkeen (WordPress 2018c, viitattu 11.4.2018).

Alatunnistetiedosto voidaan tuoda sivupohjiin funktiolla get_footer, joka toimii samalla tavalla kuin ylätunnistetiedoston tuominen get_header-funktiolla. Alatunnisteen tiedosto täytyy nimetä joko footer.php:ksi tai footer-nimi.php:ksi, sillä ilman footer-termiä, funktio tuo WordPressin mukana toimitettavan ylätunnistetiedoston (wp-includes/theme-compat/footer.php). (WordPress 2018s, viitattu 16.5.2018.)

Teemalle on määritetty footer-elementti (kuva 15), jolle on annettu luokat ”container-fluid”, ”navbar-dark” ja ”bg-dark”. Elementille on määritetty vaalea teksti tummalla taustalla luokilla ”navbar-dark” ja ”bg-dark”, mutta elementti ei kuitenkaan käytä navbar-komponenttia. Elementti saadaan levitettyä koko näytön leveydelle luokalla ”container-fluid”. Alatunniste-elementin teksti on keskitetty p-

elementin luokalla ”text-center” ja himmennetty luokalla ”text-muted”. Tekstiksi elementille haetaan WordPressiin määritetty sivuston nimi bloginfo(’name’)-funktiolla. Ennen body-tagin sulkemista määritetään wp_footer-funktio, joka sijoittaa esimerkiksi aikaisemmin määritetyt JavaScript-tiedostot kyseiseen kohtaan. Koska alatunnistetiedosto on viimeinen tiedosto, joka ajetaan sivustolla, html-tagi täytyy myös sulkea.

```
        <footer class="container-fluid navbar-dark bg-dark">
            <div class="row">
                <div class="col">
                    <p class="text-center text-muted">
                        <?php bloginfo('name'); ?>
                    </p>
                </div>
            </div>
        </footer>

        <?php wp_footer(); ?>

    </body>
</html>
```

Kuva 15. Teeman alatunniste

4.3 Sivupohjat

Teemaan voidaan kehittää useita eri sivupohjia, mutta jokaista niistä ei tarvitse toteuttaa. Sivupohjilla on oma hierarkiajärjestelmä, josta WordPress valitsee sopivimman sivupohjan näytettäväksi. (WordPress 2018c, viitattu 11.4.2018.) Esimerkiksi, jos teemaan ei määritetä mitään muuta sivupohjaa kuin ”index.php”, niin kaikkien sivujen ja artikkeleiden sisältö avautuu käyttäen index.php-tiedostoa sivupohjanaan. Mikäli teeman hakemistoon luodaan uusi tiedosto ”singular.php”, WordPress käyttää automaattisesti tätä tiedostoa sivupohjana sivuille ja artikkeleille. Tätä voidaan edelleen tarkentaa luomalla tiedosto ”single.php”, ”page.php” tai molemmat, jolloin näitä tiedostoja käytetään sivupohjina erikseen artikkeleille ja sivuille. Vieläkin tarkempi määritys on mahdollinen, kuten esimerkiksi: ”page-4.php” tiedostoa käytetään sivupohjana sivulle, jonka id-numero on 4, tai ”page-test.php”, jota käytettäisiin sivupohjana sivulle, jonka niin sanottuna slugina, eli polkutunnuksena on ”test”. Etusivulle voidaan luoda sivupohjatiedosto front-page.php, jonka WordPress automaattisesti ottaa käyttöön etusivuksi määritetylle sivulla. Virhesivupohjaksi voidaan luoda tiedosto 404.php, jonka WordPress automaattisesti näyttää, jos sivua ei löydy. (WordPress 2018b, viitattu 11.4.2018.)

WordPress näyttää sivut ja artikkelit käyttäen PHP-koodia, jolle on annettu nimeksi ”The Loop”. Koodiin sisältyy koodisilmukka, joka prosessoi WordPress-sivun tai -artikkelin sisällön näytettäväksi sivustolla. Silmukka käyttää have_posts-funktiota tarkistaakseen onko WordPressin tietokantahauilla yhtään tulosta käytäväksi läpi, jotka voidaan näyttää sivustolla muun muassa käyttämällä funktioita ”the_title”, ”the_content” ja ”the_time”. Sivun tai artikkelin otsikko saadaan funktiolla ”the_title”, sisältö funktiolla ”the_content” ja julkaisumäärä funktiolla ”the_time”, jonka päivämäärä- ja aikaformaatti voidaan määrittää lisäparametreillä, kuten esimerkiksi suomalainen päivämääräformaatti saadaan funktiolla the_time(’d.m.Y’). Sisältöä näyttävät funktiot täytyy sijoittaa silmukan sisään, jotta ne toimivat. Silmukkaan voidaan myös määrittää virheilmoitus, jos sivua tai artikkelia ei löydy. (WordPress 2018t, viitattu 16.5.2018; WordPress 2018u, viitattu 16.5.2018.)

Mikäli sivunpohjan halutaan olevan käyttäjän valittavissa sivua luodessa tai muokatessa, sivupohjatiedostoon täytyy määrittää vähintään nimi (Template Name), joka kuvaa sivupohjan käyttötarkoitusta. Oletuksena sivupohja on tarkoitettu sivuille, mutta sivupohjan voi myös määrittää artikkeleja varten määrittämällä sivupohjatiedostoon ”Template Post Typen”. (WordPress 2018w, viitattu 16.5.2018.)

Twenty Seventeen -teeman sivupohjista on sijoitettu osia template-parts-kansioon, jotta itse sivupohjatiedostot pysyvät helppolukuisina ja että mahdolliset lapsiteemojen kehittäjät voivat käyttää erillisiä osia esimerkiksi uusissa sivupohjissa tai korvata sivupohjien osat omilla osillaan. WordPressin dokumentaation mukaan (WordPress 2018x, viitattu 16.5.2018) sivupohjien osien latausta varten on get_template_part-funktio, jolla voidaan ladata sivupohjan osa itse sivupohjatiedostoon.

Vaattojärven teeman pääsivupohjatiedostoon index.php (kuva 16) on määritetty ylä- ja alatunnisteen lataaminen ja säiliö-, rivi- ja sarake-elementit, joiden sisällä suoritetaan WordPressin ”The Loop” -silmukka. Kyseisen silmukan sisällä haetaan sivun tai artikkelin otsikko ja sisältö. Lisäksi ”is_page”-funktio tarkastaa onko kyseessä sivu vai artikkeli, sillä vain artikkeleihin halutaan julkaisupäivämäärä näkyville. Funktioilla ”get_the_time(get_option('date_format'))” haetaan artikkelin julkaisupäivämäärä WordPressin asetusten mukaisessa formaatissa. Pääsivupohjan määrittämisen jälkeen teemassa toimivat sivut ja artikkelit.

```php
<?php get_header(); ?>

<div class="container">
    <?php if ( have_posts() ) : while ( have_posts() ) : the_post(); ?>
        <div class="row">
            <div class="col">

                <h2><?php the_title(); ?></h2>

                <?php if (!is_page()): ?>
                    <p class="font-weight-light">
                        <?php echo get_the_time( get_option('date_format') ); ?>
                    </p>
                <?php endif; ?>

                <?php the_content(); ?>

            </div>
        </div>
    <?php endwhile; endif; ?>
</div>

<?php get_footer(); ?>
```

Kuva 16. Teeman index.php-sivupohja

Etusivua varten on luotu tiedosto front-page.php, koska etusivun halutaan näyttävän erilaiselta kuin muut sivut, jotka käyttävät sivupohjanaan index.php-tiedostoa. Sivupohjasta on jätetty pois sivun otsikko, koska sitä ei haluta näyttää etusivulla. Lisäksi etusivulle tuodaan useita artikkeleita kategorian mukaan jaettuna.

Lisäksi teemalle on luotu virhesivupohja 404.php, jonka WordPress osaa näyttää, mikäli käyttäjä syöttää väärän osoitteen tai linkitettyä sivua ei löydy. Sivu ei tarvitse ”The Loop” -koodisilmukkaa, mutta get_header()- ja get_footer()-funktiot on määritetty. Sivu näyttää yksinkertaisen ilmoituksen ”Virhe: sivua ei löydy”.

4.4 Artikkeleiden haku sivulle kategorian mukaan

Sivupohja index.php pystyy listaamaan kaikki artikkelit kategorian mukaan, mutta artikkeleiden sijoittaminen toiselle sivulle ei onnistu hallintapaneelin kautta. On olemassa lisäosia, joilla sivun sisältö voidaan sijoittaa toiselle sivulle lyhytkoodien avulla, mutta toiselta sivulta tuodun sisällön sijoittaminen elementteihin ei onnistu.

Kun sivustolla näytetään sisältöä, ”WP_Query”-luokan objekti hoitaa sivujen ja artikkeleiden noutamisen tietokannasta ”The Loop” -koodisilmukan ajon aikana. Tätä luokkaa voidaan myös käyttää

tarvittaessa teemojen ja lisäosien kehityksessä noutamaan halutut artikkelit tai sivut. Hakusilmukka toimii samalla tavalla kuin ”The Loop”. Luokan objektille voidaan määrittää hakua rajoittavia lisäparametrejä, kuten esimerkiksi ”author” ja ”category_name”, joista ”author” rajoittaa haun vain määritettyyn kirjoittajaan ja ”category_name” rajoittaa haun vain artikkeleihin, jotka kuuluvat määritettyyn kategoriaan. (WordPress 2018v, viitattu 24.5.2018.)

Bootstrapin neljännessä versiossa esitelty Card-komponentti (kuva 17) on mukautuva ja laajennettava sisällön säiliö. Komponenttia voi käyttää esimerkiksi esikatselukuvina, listoina, ylä- ja alatunnisteina sekä yhdistää navigaatiovalikkoon. Card-komponentti tukee muun muassa kuvia, otsikoita, tekstiä, listoja ja linkkejä. (Bootstrap 2018a, viitattu 24.5.2018.)

```
<div class="container">
  <div class="row">
    <div class="card" style="width: 18rem;">
      <img class="card-img-top" src="http://
      via.placeholder.com/300x150" alt="Thumbnail">
      <div class="card-body">
        <h5 class="card-title">Otsikko</h5>
        <p class="card-text">
        Lorem ipsum dolor sit amet, consectetur adipiscing
        elit. Aenean interdum velit vitae lorem accumsan, quis
        viverra dolor tempor.
        </p>
      </div>
    </div>
  </div>
</div>
```

Kuva 17. Esimerkki card-komponentin käytöstä

Teemalle on kirjoitettu oma funktio artikkeleiden hakuun kategorian mukaan functions.php-tiedostoon (kuva 18). Funktion nimen alkuun on määritetty teeman tunniste ”vaattojarvi”, ettei toinen saman niminen funktio aiheuta sivuston kaatumista, sillä funktion voi määrittää vain kerran. Funktio vastaanottaa neljä parametriä title, category, amount ja order, eli otsikko, kategoria, määrä ja järjestys. Kategoria, määrä ja järjestys on määritetty lisäparametreiksi ”WP_Query”-luokan objektille ”the_query”. Hakusilmukan sisällä haun tulokset, kuten linkki artikkeliin, artikkelin otsikko, julkaisupäivämäärä ja sisältö on sijoitettu card-komponenttiin. Kyseistä funktiota kutsutaan muun muassa etusivulla komennolla ”vaattojarvi_echo_cards_by_category("Tieverkko ja valokuitu", "tieverkko ja valokuitu", 3)”, joka määrittää otsikoksi ja kategoriaksi ”tieverkko ja valokuitu” sekä rajoittaa artikkeleiden määrän kolmeen. Listausjärjestyksenä on oletuksena laskeva, eli uusin artikkeli ensin.

```php
function vaattojarvi_echo_cards_by_category($title, $category, $amount = -1, $order = 'DESC') {
    echo '<h2>' . $title . '</h2>';

    $args = array(
        'posts_per_page' => $amount,
        'category_name' => $category,
        'order' => $order,
        );
    $the_query = new WP_Query( $args );
    while ($the_query -> have_posts()) : $the_query -> the_post();
        echo '<div class="card"><a href="' . get_permalink() . '"><div class="card-body
            list-group-item-action text-dark">';
        echo '<h5 class="card-title">' . mb_strimwidth(get_the_title(), 0, 60, ' ...') . '</h5>';
        echo '<h6 class="card-subtitle mb-2 text-muted">' . get_the_date( 'd.m.Y' ) . '</h6>';
        echo '<p class="card-text">' . wp_trim_words( get_the_content(), 15, "..." ) . '</p>';
        echo '</div></a></div>';
    endwhile;
    wp_reset_postdata();

}
```

Kuva 18. Artikkeleiden haku kategorian mukaan

4.5 Käännösvalmius

WordPressiä käytetään ympäri maailmaa, joten teemojen ja lisäosien käännösvalmius on tärkeää. Vaikka omaa kehitettyä teemaa ei ole tarkoitus tehdä levitettäväksi, käännösvalmiuden ohjelmointi ei vie paljoa aikaa ja on hyvän WordPress-ohjelmoinnin tapaista. Jos kehitettyä teemaa tarvitaankin toisessa projektissa eri kielellä, sen kääntäminen tapahtuu helposti käännösvalmiuden johdosta.

Käännösvalmiuteen tarvitaan muun muassa __()-, esc_html__()-, _n()- ja _x()-funktioita. Funktiota ”__()” käytetään merkkijonoon, johon ei sijoitetaan muuttujia, merkkijono ei tarvitse erikseen yksikkö- ja monikkomuotoa eikä merkkijonoa ole sijoitettu HTML-koodiin. Mikäli käännettävä merkkijono sijoitetaan HTML-elementin sisälle, käytetään funktiota ”esc_html__()”, joka estää haitallisen koodin ajamisen. Funktio ”_n()” mahdollistaa yksikkö- ja monikkomuotojen käyttämisen, mikäli esimerkiksi sanojen taivutusmuoto muuttuu määrän vuoksi. Funktiolla ”_x()” voidaan helpottaa kääntämistä kommentoimalla, mihin käännettävää merkkijonoa käytetään. Funktiot palauttavat merkkijonon, mutta mikäli merkkijono halutaan tulostaa, käytetään erillisiä e-funktioita, kuten esimerkiksi _e(), esc_html_e() ja _ex(). Funktiota ”_n()” käytetään yleensä yhdessä ”printf()”-funktion kanssa, jotka mahdollistavat muuttujan sijoittamisen merkkijonoon ja yksikkö- tai monikkomuodon käyttämisen muuttujan mukaan. Funktiot tarvitsevat myös lisäparametrin, jossa määritetään käännösten ”text domain”, eli mille teemalle tai lisäosalle käännös kuuluu. (WordPress 2018h, viitattu 2.5.2018.) Mikäli lisäparametriä ei määritetä, WordPress yrittää etsiä käännöstä omista käännöstiedostoista. Esimerkiksi funktio _e(’Title’) tulostaisi ”Otsikko”, mikäli WordPressissä on käytössä Suomen kieli.

Vaattojärven teemassa suurin osa merkkijonoista haetaan WordPressin omilla funktioilla, joten sivupohjissa ei ole paljoa käännettävää. Tiedostossa 404.php oli käännettävänä otsikko ”virhe” ja ”etsimääsi sivua ei löydy”, jotka on sijoitettu esc_html_e()-funktioihin ja text domain -lisäparametriksi on määritetty ”vaattojarvi”. Lisäksi sivupohjaan index.php julkaisupäivämäärä saadaan tukemaan käännöksiä ja käyttäjän määrityksiä funktiolla ”date_i18n(get_option('date_format'))”.

5 POHDINTA

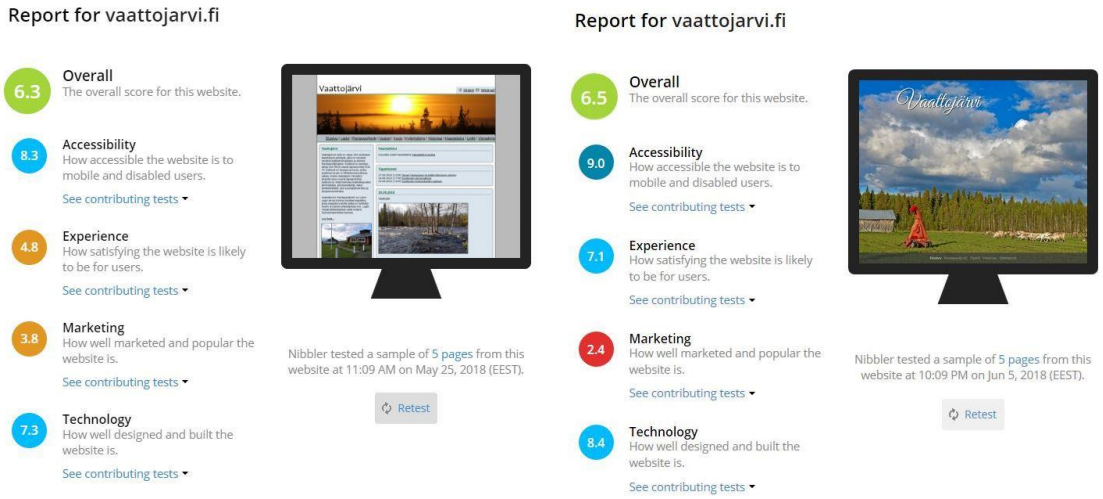
Tämän opinnäytetyön tavoitteena oli kehittää responsiiviset verkkosivut Vaattojärven kyläyhdistykselle käyttäen WordPress-sisällönhallintajärjestelmää. Valmis sivusto on tavoitteiden mukainen, mutta sen saavuttaminen vaati enemmän aikaa ja tarkempaa tutustumista WordPressin dokumentaation kuin osasin odottaa. WordPress on todella monipuolinen sisällönhallintajärjestelmä ja tuntuu, että olen vasta päässyt kunnolla tutustumaan siihen tämän opinnäytetyön aikana, vaikka olen tehnyt jo aikaisemmin useita projekteja käyttäen WordPressiä.

Opin paljon uutta WordPress-kehityksestä tämän opinnäytetyön aikana, kuten mukauta-valikon toiminnasta, teemojen ja lisäosien kääntämisestä ja erityisesti lisäosien kehityksestä, vaikka sitä ei tässä opinnäytetyössä käsiteltykään. Halusin luoda hallintapaneeliin selkeämmän rakenteen, joten päädyin opiskelemaan Udemyn kurssin kautta WordPress-lisäosien kehittämistä. Sen sijaan, että kaikki uutiset, tapahtumat ja tanssitapahtumat olisi kirjoitettu artikkeleina ja eroteltu kategorioiden avulla, päädyin luomaan uudet kustomoidut sisältötyypit (custom post type) jokaista varten. Nyt hallintapaneelissa on selkeästi eroteltuna uutiset, tapahtumat ja tanssitapahtumat omina sisältötyyppeinä, joihin voitiin sisällyttää lisäominaisuuksia Advanced custom fields -lisäosan avulla, kuten esimerkiksi kalenteri tapahtumapäivämäärää varten.

Haasteita toivat paikoitellen WordPressin heikko dokumentointi. Esimerkiksi oikeaoppiseen WordPress-kehitykseen oli välillä vaikea löytää oikeaa ratkaisua, sillä dokumentaatiossa joistakin asioista ei mainita ollenkaan ja hakukoneella etsiessä eri kehittäjien ratkaisua ongelmaan, suurin osa hakutuloksista on useita vuosia vanhoja. Bootstrapin käytössä en kokenut suurempia haasteita, mutta neljännestä versiosta on poistettu tai muutettu useista elementeistä marginaalin (margin) ja täytteen (padding) määrää, jolloin elementit ovat todella lähellä toisiaan ja paremman käyttökokemuksen saavuttamiseksi ne täytyi erottaa toisistaan. Uudessa versiossa on kuitenkin myös luokkia marginaalin ja täytteen lisäämiseen elementeille, mutta mielestäni olisi ollut parempi vaihtoehto jättää elementeille oletuksena suurempi marginaali. Uusi breakpoint oli tervetullut muutos, mutta olisin kaivannut vielä suurempaa vaihtoehtoa xl-breakpointille, sillä varsinkin suuremmille 4K-resoluution näytöille mahtuisi paljon enemmän sisältöä vierekkäin. Breakpointien leveyttä voidaan kuitenkin muuttaa SASS-muuttujien (Syntactically Awesome Style Sheets) avulla tarpeen tullen. Vaikka Bootstrap 4 tukee Internet Explorer 10 -selainta, itse selain ei tue esimerkiksi Flexbox-moduulia, joten sivuston toimivuudesta selaimella ei voida olla varmoja. Tosin kyseisen selaimen

käyttö on nykyään todella vähäistä ja Vaattojärven sivuston kävijämäärän perusteella ei ole oleellista tukea yli 6 vuotta vanhaa selainta.

Sivustolta jäivät vielä puuttumaan vieraskirja ja haastattelut-sivu, jotka tullaan vielä kehittämään myöhemmässä vaiheessa. Toimeksiantajan ja käyttäjien arvioit sivustosta ovat olleet positiivisia, sillä sivuston ulkoasu näyttää hyvältä, on helppokäyttöinen ja sitä on helppo ylläpitää. Testasin sivuston myös Nibbler-sivustolla (kuva 19), jonka avulla voidaan testata verkkosivustoja. Helppokäyttöisyyden (accessibility), kokemuksen (experience) ja teknologian (technology) pisteet nousivat, mutta markkinointi (marketing) vetää yleistä pisteytystä alaspäin. Tämä johtuu luultavasti vielä heikosta hakutuloksesta, sillä sivustolla otettiin ensimmäistä kertaa SSL-suojaus käyttöön, joten esimerkiksi Google ei vielä listaa uutta sivustoa ollenkaan, vaan linkit osoittavat vanhaan sivustoon.



Kuva 19. Nibbler-sivustotestaajan (Nibbler 2018, viitattu 8.6.2018) arvosanat vanhalle ja uudelle sivustolle

Teeman osalta olen harkinnut, että jatkokehittäisin siitä enemmän kustomoitavan teeman, joka olisi nopeasti muokattavissa useaan käyttötarkoitukseen. Jatkokehitetty teema tulisi joko pelkästään omaan käyttööni tai ostettavaksi pientä korvausta vastaan. Mikäli päätän jakaa teemaa, koodi täytyy dokumentoida, varmistaa teeman toiminta lapsiteemana sekä monipuolistaa, sillä teemaa voidaan tulla käyttämään todella moneen erilaiseen sivustoon, joista jokaisella on erilaiset tarpeet.

Mielipiteeni on vain vahvistunut positiivisesti siitä, että WordPress on hyvä ja mukautuva sisällönhallintajärjestelmä, jota tulen varmasti käyttämään vielä tulevaisuudessa uusien projektien kanssa. Ainoa negatiivinen asia, joka WordPressistä tulee mieleen, on sen suuri opiskelukynnys. Uudelle

kehittäjälle WordPress on suuri haaste, joka vaatii harrastuneisuutta, jota voitaisiin helpottaa luomalla parempia oppaita kehittäjien saataville, joissa käsitellään enemmän käytännön esimerkkejä. Oman teeman kehitys on suuren opiskelukynnyksen takana, mutta varmistaa sen, että sivusto sisältää kaikki tarvittavat ominaisuudet.

LÄHTEET

Bootstrap 2018a. Cards. Viitattu 24.5.2018, https://getbootstrap.com/docs/4.1/components/card/

Bootstrap 2018b. Migrating to v4. Viitattu 10.4.2018, https://getbootstrap.com/docs/4.1/migration/

Bootstrap 2018c. Introduction. Viitattu 23.4.2018, https://getbootstrap.com/docs/4.1/getting-started/introduction/

Bootstrap 2018d. Grid system. Viitattu 23.4.2018, https://getbootstrap.com/docs/4.0/layout/grid/

Bootstrap 2018e. Spacing. Viitattu 24.4.2018, https://getbootstrap.com/docs/4.1/utilities/spacing/

Bootstrap 2018f. Contents. Viitattu 1.5.2018, https://getbootstrap.com/docs/4.1/getting-started/contents/

Bootstrap 2018g. Navbar. Viitattu 12.5.2018, https://getbootstrap.com/docs/4.1/components/navbar/

Google 2018. Best practices for mobile-first indexing. Viitattu 25.5.2018, https://developers.google.com/search/mobile-sites/mobile-first-indexing

Nibbler 2018. Report for vaattojarvi.fi. Viitattu 8.6.2018, http://nibbler.silktide.com/en_US/reports/vaattojarvi.fi

W3C 2017. CSS Flexible Box Layout Module Level 1. Viitattu 24.5.2018, https://www.w3.org/TR/css-flexbox-1

WP Bootstrap Navwalker. WP Bootstrap Navwalker. Viitattu 14.5.2018, https://github.com/wp-bootstrap/wp-bootstrap-navwalker

jQuery 2018. Downloading jQuery. Viitattu 1.5.2018, http://jquery.com/download/

Obénland, K. 2014. Title Tags in 4.1. Viitattu 3.5.2018, https://make.wordpress.org/core/2014/10/29/title-tags-in-4-1/

Soltano, S. 2018. Web technology fact of the day. Viitattu 1.4.2018, https://w3techs.com/blog/entry/fact_20180305

Statcounter 2018. Desktop vs Mobile vs Tablet Market Share Worldwide. Viitattu 25.5.2018, http://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/worldwide

Vaattojärvi 2018. Vaattojärven kylän historiaa. Viitattu 1.4.2018, http://vaattojarvi.fi/historiaa

W3Schools 2018a. CSS Flexbox. Viitattu 4.4.2018, https://www.w3schools.com/CSS/CSS3_flex-box.asp

W3Schools 2018b. HTML <!DOCTYPE> Declaration. Viitattu 12.4.2018, https://www.w3schools.com/tags/tag_doctype.asp

W3Schools 2018c. HTML Encoding (Character Sets). Viitattu 16.4.2018, https://www.w3schools.com/html/html_charset.asp

W3 Techs 2018. Usage of content management systems for websites. Viitattu 1.4.2018, https://w3techs.com/technologies/overview/content_management/all/

WordPress 2018a. Organizing Theme Files. Viitattu 11.4.2018, https://develo-per.wordpress.org/themes/basics/organizing-theme-files/

WordPress 2018b. Template Hierarchy. Viitattu 11.4.2018, https://develo-per.wordpress.org/themes/basics/template-hierarchy/

WordPress 2018c. Theme Development. Viitattu 11.4.2018, https://co-dex.wordpress.org/Theme_Development

WordPress 2018d. Code Reference: wp_enqueue_style. Viitattu 12.4.2018, https://develo-per.wordpress.org/reference/functions/wp_enqueue_style/

WordPress 2018e. Function Reference: language attributes. Viitattu 12.4.2018, https://codex.wordpress.org/Function_Reference/language_attributes

WordPress 2018f. Code Reference: bloginfo. Viitattu 16.4.2018, https://developer.wordpress.org/reference/functions/bloginfo/

WordPress 2018g. Theme Tags. Viitattu 2.5.2018, https://make.wordpress.org/themes/handbook/review/required/theme-tags/

WordPress 2018h. I18n for WordPress Developers. Viitattu 2.5.2018, https://codex.wordpress.org/I18n_for_WordPress_Developers

WordPress 2018i. Function Reference: wp_enqueue_script. Viitattu 2.5.2018, https://developer.wordpress.org/reference/functions/wp_enqueue_script/

WordPress 2018j. Theme Functions. Viitattu 2.5.2018, https://developer.wordpress.org/themes/basics/theme-functions/

WordPress 2018k. Function Reference: get header. Viitattu 3.5.2018, https://codex.wordpress.org/Function_Reference/get_header

WordPress 2018l. Function Reference: wp_title. Viitattu 3.5.2018, https://developer.wordpress.org/reference/functions/wp_title/

WordPress 2018m. Filter Reference: body class. Viitattu 4.5.2018, https://codex.wordpress.org/Plugin_API/Filter_Reference/body_class

WordPress 2018n. Customizer Objects. Viitattu 5.5.2018, https://developer.wordpress.org/themes/customize-api/customizer-objects/

WordPress 2018o. Function Reference: is front page. Viitattu 6.5.2018, https://codex.wordpress.org/Function_Reference/is_front_page

WordPress 2018p. Function Reference: wp_nav_menu. Viitattu 12.5.2018, https://developer.wordpress.org/reference/functions/wp_nav_menu/

WordPress 2018q. Hook Reference: nav_menu_css_class. Viitattu 12.5.2018, https://developer.wordpress.org/reference/hooks/nav_menu_css_class/

WordPress 2018r. Class Reference: Walker. Viitattu 14.5.2018, https://codex.wordpress.org/Class_Reference/Walker

WordPress 2018s. Function Reference: get_footer. Viitattu 16.5.2018, https://codex.wordpress.org/Function_Reference/get_footer

WordPress 2018t. The Loop. Viitattu 16.5.2018, https://codex.wordpress.org/The_Loop

WordPress 2018u. Function Reference: have posts. Viitattu 16.5.2018, https://codex.wordpress.org/Function_Reference/have_posts

WordPress 2018v. Class Reference: WP_Query. Viitattu 24.5.2018, https://codex.wordpress.org/Class_Reference/WP_Query

WordPress 2018w. Page Templates. Viitattu 16.5.2018, https://developer.wordpress.org/themes/template-files-section/page-template-files/

WordPress 2018x. Function Reference: get_template_part. Viitattu 16.5.2018, https://developer.wordpress.org/reference/functions/get_template_part/